Combinatorial Complexity and Tree Modelling in the Analytical Framework of Hi-Lo Equation Strategy

Attara Majesta Ayub- 13552139
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522139@itb.ac.id

***Abstract*—The Hi-Lo Equation introduces a distinctive gaming concept wherein players employ numerical cards and arithmetic operations to formulate equations that approximate the values of 1 or 20. Through strategic employment of mathematical operations—addition, subtraction, division, multiplication, and even square roots—players craft an equation approaching the chosen target value. This paper will analyze the game mechanics with discrete mathematical concepts, particularly combinatorics and tree, to breakdown the optimal outcomes based on card distribution among players, thereby refining their gameplay strategies.**

***Keywords*—Hi-Lo Equation, Combinatorics, The Devil's Plan, Game Strategy**

I. INTRODUCTION

"The Devil's Plan" showcases a seven-day reality gaming show aimed at evaluating contestants' intellectual prowess, focusing on cognitive skills rather than physical abilities. This distinctive show, integrated into the broader program of "The Devil's Plan," sets itself apart from digital gaming by emphasizing survival gaming and promoting collaboration and face-to-face communication. Among the challenges presented, the Equation Hi-Lo game stands out, blending mathematical concepts with card gaming strategies. Equation Hi-Lo creatively combines elements from Poker, Hi-Lo, and the mathematical notion of 24. It shifts away from chance-based games, engaging players in strategic problem-solving.

Fig 1.1) Shows the cards used in Equation Hi-Lo (Taken from Netflix)

Equation Hi-Lo, though rooted in poker, surpasses mere chance-driven gameplay. It requires intricate calculations, strategic planning, mathematical skills, guiding players through successive rounds. This endurance-demanding game culminates in selective elimination, posing an intense, repetitive equation-solving challenge testing contestants' mental resilience until defeat.

In the Hi-Lo Equation game, combinatorics and permutations are crucial for strategic gameplay, intertwined with math and decision-making. Permutations can be used to explore various combinations of number cards and math operations players hold. Combinatorics, primarily showcased through iterating possible permutations of numeric cards and operation symbols, enables the calculation of diverse equations that lead to the target values of a number close to 1 or 20. This mathematical concept could help players predict outcomes and pick smart moves based on card combos and strategy. An algorithm is essential to show how players can strategically select the best combinations. The algorithm should also utilize tree implementation, maximizing strategic calculations and enhancing gameplay depth, rather than just luck.

II. COMBINATORICS THEORY

Combinatorics is a branch of mathematics that deals with counting the number of possible arrangements or selections of objects without needing to enumerate all possibilities. While enumeration might suffice for smaller scenarios, as the number of elements grows, this method becomes impractical. Combinatorial Theory steps in, offering efficient ways to solve such problems.

The Rule of Product is used when multiple independent events are to be combined. If there are P outcomes in the first event and Q outcomes in the second event, the total outcomes when both events occur are $P \times Q$. For instance, selecting representatives where there are 65 men and 15 women in a group involves choosing 1 man and 1 woman, resulting in $65 \text{ x } 15 = 975$ possible combinations. On the other hand, *The Rule of Sum* applies when there are choices that are mutually exclusive or cannot occur together; you can sum the number of outcomes of each choice. Consider selecting a leader from the same group of 65 men and 15 women. The total options would be $65 + 15 = 80$ for choosing a leader.

The principle of inclusion and exclusion helps us calculate number of elements in the union of certain sets. It considers set sizes and their intersections. Suppose that you have two sets A and B. The size of the union is at most $|A| + |B|$, however, we are double counting all elements in $A \cap B$. Therefore, the formula

$$|A \cup B| = |A| + |B| - |A \cap B|$$

correcting the overlapping elements. This principle extends to multiple sets, allowing us to calculate the union by considering the sizes and intersections of various subsets of these sets.

Permutations represent the number of different arrangements of objects. Permutations are used when the order of arrangement matters. $P(n,r)$ stands for permutations of r elements from a set of n elements.

Formula:

$$P(n,r) = \frac{n!}{(n-r)!}$$

Permutation notation:

$$P(n,r) = \binom{n}{r} = {}_nP_r$$

Combinations refer to the number of selections of items where the order doesn't matter. $C(n,r)$ represents combinations of r elements from a set of n elements.

Formula:

$$C(n,r) = \frac{n!}{r! \times (n-r)!}$$

Combination notation:

$$C(n,r) = \binom{n}{r} = {}_nC_r$$

Combinations are used when the order of arrangement is irrelevant. Such as determining the number of possible passwords given specific criteria (length, characters allowed), calculating the number of arrangements for seats in a cinema row for different scenarios: well-lit vs. dark conditions, forming committees, selecting representatives from different groups or pools of individuals, considering various constraints (gender, department, etc.).

Expanding on the fundamental principles of combinatorics, the *Extended Rule of Multiplication* applies to n experiments, each with distinct outcomes $p_1, p_2, \ldots, p_n$. Conducting all n experiments leads to a combined count of $p_1 \times p_2 \times \ldots \times p_n$ outcomes. For instance, determining the count of odd numbers between 1000 and 9999 involves four experiments for each digit position. Calculating these yields 5 x 8 x 8 x 7 = 4500 odd numbers, showcasing the application of this principle in complex counting scenarios.

Similarly, the *Extended Rule of Addition* involves n experiments, each offering outcomes $p_1, p_2, \ldots, p_n$. In this case, the total outcomes become $p_1 + p_2 + \cdots + p_n$. This principle finds utility in various scenarios, such as arranging balls into boxes, where different permutations emerge from these varied outcomes.

The final formula for the *extended Rule of Multiplication* is given by:

$$p_1 \times p_2 \times \ldots \times p_n$$

And for the *extended Rule of Addition*:

$$p_1 + p_2 + \cdots + p_n$$

III. TREE THEORY

A tree is a specific type of undirected graph that is both connected and acyclic. Connectedness implies that a tree is connected, meaning that there exists a path between any pair of vertices in the graph. Acyclic means a tree does not contain any cycles, ensuring that there are no closed loops or repeated edges between vertices. In a tree with n vertices, the number of edges is always $(n-1)$. This is a foundational principle. A forest is essentially a collection of trees that are not connected to each other.

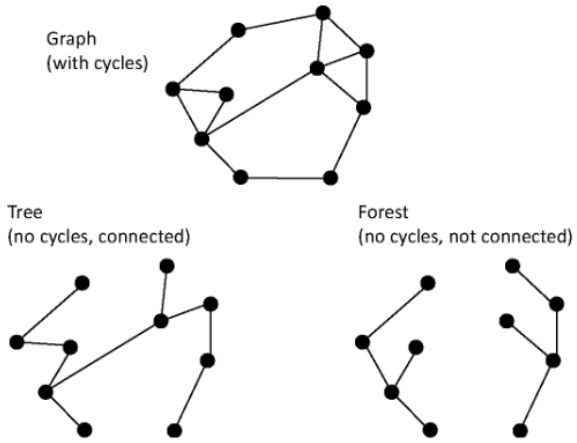

Fig 3.1) Example of graph, trees, and forest (Taken from Rinaldi Munir – Pohon "Bag. I")

A graph is a tree if and only if it is minimally connected.

Proof. Let *G* be a graph.

To prove the direct implication, assume *G* is a tree and consider a contradiction where *G* isn't minimally connected. This implies that within *G*, there exists an edge $e = vw$ that, upon removal, results in a connected graph denoted as *G'*. Because *G'* maintains connectivity, there exists a sequence $v_1v_2\ldots v$ in *G'* that forms a path connecting v to w, notably with v_1 as v and v as w. Assume that *G* is a tree and suppose, towards a contradiction, that *G* is not minimally connected. Then G contains an edge $e = vw$ that we can delete to obtain a connected graph G'. Since G' is connected, there must be a

path $v_1v_2\ldots v'$ in G' connecting v to w (in particular, $v_1 = v$ and $v' = w$).

Given that $vw \notin V(G')$, the sequence $v_1v_2\ldots v'v_1$ forms a cycle within G, creating the necessary contradiction. To demonstrate the reverse scenario, let's suppose G is minimally connected and consider a contradiction where G isn't a tree. This implies that G contains a cycle $v_1v_2\ldots v'v_1$. Let G' represent the graph obtained from G by eliminating the edge v_1v_2. We claim that G' is a connected graph. To prove this, let v and w be two distinct vertices of G', and let P be a path in G from v to w, which must exist because G is connected. If v_1v_2 is not an edge of P, then P is a path from v to w in G'. Otherwise, we can replace the edge v_1v_2 in P by the path $v_1v_lv_{l-1}\ldots v_2$ to turn P into a walk from v to w in G'. Among all such walks, pick one P_0 of minimum length. It is clear that P_0 is a path from v to w in G. Hence, we conclude that G' is connected, which contradicts that G is minimally connected.

There are various types of trees,

1. Rooted Trees: These are trees where one vertex is distinguished as the "root." Each edge has a direction away from the root, creating a hierarchical structure.
2. Binary Trees: A specific type of rooted tree where each vertex has at most two children.
3. N-ary Trees: A rooted tree in which each node has no more than N children

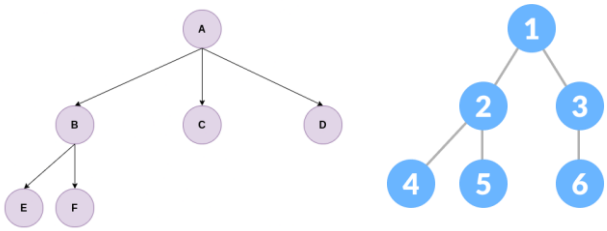

Fig 3.2) Example of n-ary and binary tree (Taken from Algorithm Tutor)

There are three ways to read a binary tree:

1. Prefix: Root node - left child - right child
2. Infix: Left child - root node - right child
3. Postfix: Left child - right child - root node

For instance, consider the expression represented by a tree:

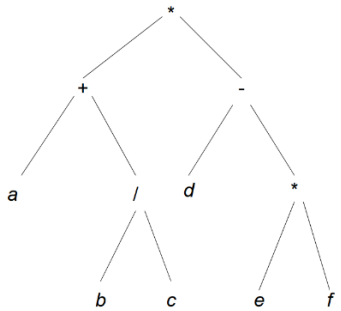

Fig 3.3) Tree Sample (Taken from Rinaldi Munir – Pohon "Bag 2.")

The string representation of an expression changes based on how it's structured:

Preorder	: * + a / b c – d * e f	(*prefix*)
Inorder	: a + b / c * d – e * f	(*infix*)
Postorder	: a b c / + d e f * - *	(*postfix*)

IV. GAME RULES

All players receive one set of symbol cards, including addition, subtraction, and division cards. There are four types of number cards: Gold, Silver, Bronze, and Dirt, each ranging from zero to ten. There are 4 cards for each number, totaling 44 cards. Gold holds the highest value, followed by Silver, Bronze, and Dirt in descending order. Additionally, four square root and multiplication cards will be used in addition to the number cards.

At the game's onset, all players must place a bet by paying one chip. Then, the dealer shuffles the 52 cards and deals one face-down card to each player, which remains hidden. If a player receives a square root or multiplication card as the hidden card, they must return it and repeat until they receive a number card. Subsequently, the dealer distributes two additional cards to each player, these are open cards visible to other players.

If a player receives a square root card as their open card, they'll receive an additional number card. However, if they receive a multiplication card, they must discard either an addition, subtraction, or multiplication card from their hand and draw one more number card. Division cards cannot be discarded. Should a player receive a square root or multiplication card as their open card and then receive another symbol card, they must discard the new symbol card and redraw until they receive a number card.

After each player receives two open cards, the first betting round begins. The game continues until all players either bet the same amount or fold, opting out of the round. Bets cannot surpass the chip count of the player with the least amount of chips. Subsequently, each player receives one more open number card. Once all open cards are distributed, players use their number and symbol cards to create a number close to either one or twenty. The square root card may only be used on one number. Following the formation of equations, the final betting phase commences, starting with the initial player.

Once the final bet concludes, players decide to bet high or low. High betting involves choosing the 20 mark and the player with the number closest to 20 wins. Low betting requires creating an equation resulting in a number closest to 1 and choosing the 1 mark. If a player can create two equations, one closest to 20 and the other closest to 1, they may place a swing bet. A player selecting a swing bet wins only if they succeed in both high and low betting. Once all players decide, markers (1 or 20) and equations are revealed, and players with the same markers compete.

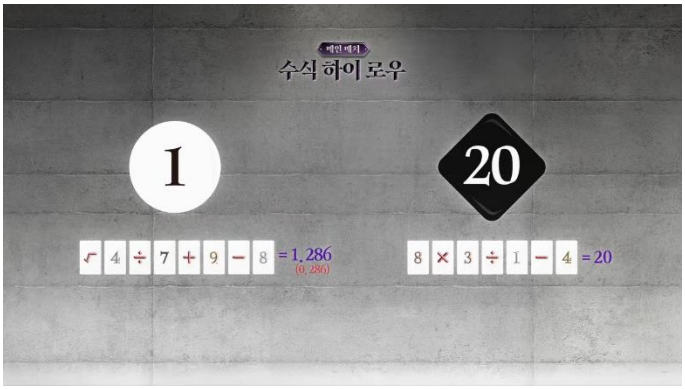

Fig 4.1) Example of equation and mark (Taken from Netflix)

Each winner of high or low bets receives half of the total chip bet collected previously. If the chips collected are an odd number, the remaining chip is discarded. In case of ties, the player with the highest number card among those who bet high wins, and among those who bet low, the player with the lowest number card wins. In the event of tied highest or lowest number cards, the winner is determined by the color rank (Gold > Silver > Bronze > Dirt). A player with 0 chip will be eliminated from the game.

V. THEORY IMPLEMENTATION

To strike the best move in the gameplay with the uncertainty of one number card, a program can be a helping hand. The logic is to employ permutations to find the best equations. The process involves generating permutations of numeric cards and operation symbols for each player. It utilizes the **next_permutation** function to generate these combinations. Within the permutation loops, the code checks whether the player possesses a square root card. For those players who received a square root symbol, the program considers every number paired with the square root operation to calculate potential equation results. It evaluates various combinations of numbers and symbols to determine the equation that yields values closest to 1 or 20, capturing this information as the best move for each player. The program finally displays the closest equations to 1 and 20 for each player, providing insight into the most favorable strategies given their available cards and symbols.

A crucial part of the algorithm is the way computers calculate equations. Computers typically perform calculations following a specific sequence or order called "left-to-right" evaluation. This means they solve expressions or equations by processing them from the leftmost character to the right, executing operations in the order they appear, unlike mathematical conventions that adhere strictly to the rules of operator precedence (like PEMDAS/BODMAS). For instance:

```
2 + 3 * 4 - 1
```

In an expression like this, a ”left-to-right” evaluation would first compute 2 + 3, resulting in 5, and then multiply 5 by 4 to get 20, then subtract by 1 resulting 19, whereas mathematical rules would prioritize multiplication, yielding 13. This distinction in calculation order often leads to different results between human mathematical expectations and computer-based evaluations.

Edsger Dijkstra created the "Shunting Yard" algorithm to convert infix expressions into postfix notation. This method employs a stack, but here, the stack stores operators rather than numbers. The stack's purpose is to reorder operators within the expression and serve as a repository, delaying operator output until both operands are available.

```
#Example 1

A * ( B + C )  -->  A B C + *

  Current Symbol     Operator Stack    Postfix String
1       A                                A
2       *                  *             A
3       (                  * (           A B
4       B                  * (           A B
5       +                  * ( +         A B
6       C                  * ( +         A B C
7       )                  *             A B C +
8                                        A B C + *
```

```
#Example 2

A - B + C  -->  A B - C +

  Current Symbol     Operator Stack    Postfix String
1       A                                A
2       -                  -             A
3       B                  -             A B
4       +                  +             A B -
5       C                  +             A B - C
6                                        A B - C +
```

Guidelines for “Shunting Yard” algorithms:

- For an incoming operand, print it.
- When an incoming symbol is a left parenthesis, push it onto the stack.
- Upon encountering a right parenthesis: discard it, pop and print symbols from the stack until a left parenthesis is found. Discard the left parenthesis.
- If the incoming symbol is an operator and the stack is empty or holds a left parenthesis at the top, push the incoming operator onto the stack.
- If the incoming symbol is an operator with higher precedence than the top stack operator or shares the same precedence and is right associative, push it onto the stack.
- When the incoming symbol has lower precedence than the top stack operator or shares the same precedence and is left associative, continue popping the stack until the condition is false. Then, push the incoming operator.
- At the expression's end, pop and print all operators from the stack (no remaining parentheses).

Further elaborating the application in the program, **shunting_yard** function is a *tree* implementation called to perform calculations based on the generated permutations.

Shunting Yard Algorithm:

```
  float shunting_yard(float num[], char
symb[])
  {
      deque<float> x;
      deque<char> y;

      for(int i = 0; i < 3; i++)
      {
          if(symb[i] == 'x')
          {
              float res = num[i]*num[i+1];
              if(i == 2)
              {
                  x.push_back(res);
              }
              else num[i+1] = res;
          }
          else if(symb[i] == ':')
          {
              float res = num[i]/num[i+1];
              if(i == 2)
              {
                  x.push_back(res);
              }
              else num[i+1] = res;
          }
          else if(symb[i] == '-' || symb[i]
== '+')
          {
              if(i == 2)
              {
                  x.push_back(num[i]);
x.push_back(num[i+1]);
                  y.push_back(symb[i]);
              }
              else
              {
                  x.push_back(num[i]);
                  y.push_back(symb[i]);
              }
          }
      }
      float ans;
      while(!y.empty())
      {
          char sign = y.front();
y.pop_front();
          float a = x.front();
x.pop_front();
          float b = x.front();
x.pop_front();
          ans = (sign == '+')? (a+b) : (a-
b);
          x.push_front(ans);
      }
      return ans;
  }
```

The code's functions and nested loops systematically explore all potential permutations, evaluating and comparing equations to determine the optimal moves for each player. The output showcases the best equations identified for achieving values nearest to 1 and 20, considering different scenarios and symbol combinations.

VI. PROGRAM EXECUTION

The simplified simulation program is created to help better understanding toward the game. In the simulation, the program begins by prompting the user to input the number of players, username, and forced bet.

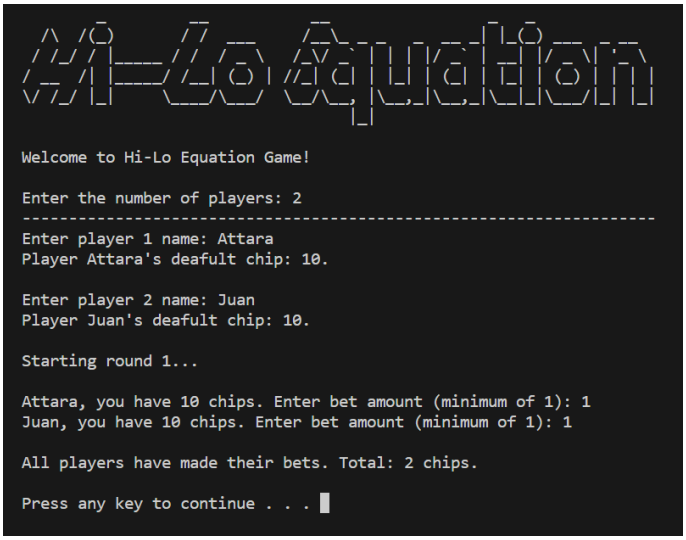

Fig 6.1) Main program (self-source)

After the cards are dealt (prior to the final bet), the player can view their card and other player's open card. Player is asked to input the equation.

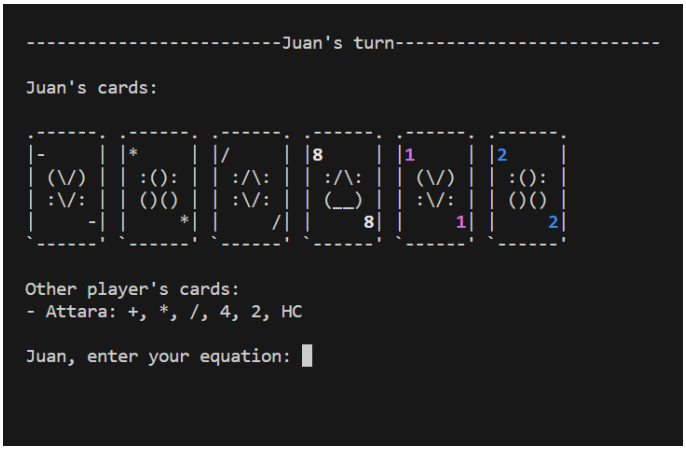

Fig 6.2) Player's turn to craft equation (self-source)

Here, the color value is replaced as follows:

Gold	->	Blue
Silver	->	Magenta
Black	->	Tosca
Dirt	->	White

Now, it is time to execute the mathematical tool that enhance "Equation Hi-Lo" players gameplay. The program begins by prompting the user to input the number of players and then proceeds to collect the player data: names, number

values, and corresponding symbols.

```
===== HI-LO PROBABILITIES =====
Enter the number of players:  2

INPUT PLAYER-1'S DATA
Input card 1 : 8
Input card 2 : 1
Input card 3 : 2
Input symbol 1 (except for sqrt card) : +
Input symbol 2 (except for sqrt card) : *
Input symbol 3 (except for sqrt card) : /
Do player have square root card? (y/n) : n

INPUT PLAYER-2'S DATA
Input card 1 : 4
Input card 2 : 2
Input card 3 : 1
Input symbol 1 (except for sqrt card) : +
Input symbol 2 (except for sqrt card) : *
Input symbol 3 (except for sqrt card) : /
Do player have square root card? (y/n) : n
```

Fig 6.3) Hi-Lo Probabilities program inputs (self-source)

Then, the program will calculate all the possibilities by using the algorithm in previous chapter, resulting in this informative output for players:

```
BEST MOVE FOR PLAYER -1

CLOSEST NUMBER TO 1 : 1
WITH DIFFERENCE : 0
IF THE LAST CARD : 0
USING EQUATION: 8 * 1 + 2 / 0

CLOSEST NUMBER TO 20 : 20
WITH DIFFERENCE: 0
IF THE LAST CARD : 9
USING EQUATION : 8 / 2 + 9 * 1
```

Fig 6.4) Best move calculation for Player 1 (self-source)

```
BEST MOVE FOR PLAYER -2

CLOSEST NUMBER TO 1 : 1
WITH DIFFERENCE : 0
IF THE LAST CARD : 0
USING EQUATION: 2 * 4 / 0 + 1

CLOSEST NUMBER TO 20 : 20
WITH DIFFERENCE: 0
IF THE LAST CARD : 6
USING EQUATION : 4 * 6 + 1 / 2
```

Fig 6.5) Best move calculation for Player 2 (self-source)

Once players receive this information, their decisions are based on calculations rather than relying solely on luck. Players can input the equation and bet mark right away.

```
HIGH BETTER
------------------------------------------------------------
- Juan, 4.667

WINNER      : Juan
DIFFERENCE  : 4.667

Juan get half of the total bet chip.
Juan's current chip: 9 + 1 = 10.

LOW BETTER
------------------------------------------------------------
- Attara, 0

WINNER      : Attara
DIFFERENCE  : 0

Attara get half of the total bet chip.
Attara's current chip: 9 + 1 = 10.

-----------------------CURRENT STATS-----------------------
Player 1    : Juan
Chips       : 10 chips

Player 2    : Attara
Chips       : 10 chips

Press any key to continue . . .
```

Fig 6.6) End of round after dealer distributed the final card (self-source)

VII. Conclusion

This paper delves into permutations and combinations in gaming strategy. It explores the role of the Shunting Yard algorithm in computational evaluations, which is incorporated into a simulation program suggesting optimal moves based on card distributions. Through this exploration, it illustrates how mathematics serves as a fundamental element applicable to diverse scenarios, including the dynamics of the Hi-Lo Equation game. By intertwining theory and practical application, the paper highlights the intersection of mathematics and strategy within the dynamics of the Hi-Lo Equation game.

VIII. Attachment

Github link to the source code Hi-Lo Equation Probabilities

IX. Acknowledgment

I would like to sincerely express my gratefulness to Allah SWT. that this paper can be completed well. Then, I would like to express my deepest gratitude to Mr. Rinaldi Munir, as my Discrete Mathematics Lecturer, for all the guidance, kindness, and fun lecture. Mas Juan, which makes me love mathematics even more, and my family and friends, especially K-03 Teknik Informatika 2022, for the endless support.

References

[1] Rinaldi Munir, Bahan Kuliah IF2120 Matematika Diskrit – Kombinatorial (Bagian 1)," Institut Teknologi Bandung. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/17-Kombinatorial-Bagian1-2023.pdf

[2] Rinaldi Munir, "Bahan Kuliah IF2120 Matematika Diskrit – Kombinatorial (Bagian 2)," Institut Teknologi Bandung. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/18-Kombinatorial-Bagian2-2023.pdf

[3] Fox Jacob "MAT 307: Combinatorics," Massachusetts. Available: https://math.mit.edu/~fox/MAT307-lecture04.pdf
[4] Felix Gotti "MIT 18.211: Combinatorial Analysis and Introduction to Tree". Available: https://math.mit.edu/~fgotti/docs/Courses/Combinatorial%20Analysis/23.%20Intro%20to%20Trees/Intro%20to%20Trees.pdf
[5] https://math.oxford.emory.edu/site/cs171/shuntingYardAlgorithm/
[6] Kemchanin Pornpipatsakul, "Win's Story (22E): The Devil's Plan — An analysis of six physical survival games through the lens of a Game Designer". Available: https://medium.com/in-game-by-memore/wins-story-22e-the-devil-s-plan-an-analysis-of-six-physical-survival-games-through-the-lens-b2a09c4cd336

STATEMENT

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2023

Attara Majesta Ayub 13522139